

[Inicio](#) | [Fundamentos](#) | [Instalación](#) | [Añadir Elementos](#) | [Encadenar Métodos](#) | [Asociar Datos](#) | [Usando sus Datos](#) | [Desplegar DIVs](#) | [La Función data\(\)](#) | [Introducción a SVG](#) | [Despliegue de SVG](#) | [Tipos de Datos](#) | [Diagrama de Barras](#) | [Diagrama de Dispersión](#) | [Escala](#) | [Ejes](#) |

Introducción de SVG

D3 es de mayor utilidad cuando se usa para generar información gráfica en formato SVG. Si bien es factible desplegar gráficas con etiquetas `div` y otros elementos nativos de HTML, este método no es el ideal y está sujeto a las inconsistencias que existen entre los diferentes navegadores. El uso de SVG es bastante más confiable, consistente visualmente y más rápido.

Aplicaciones de dibujo vectorial como Illustrator se pueden usar para generar archivos en formato SVG, pero es necesario aprender cómo se generan a partir de código.

El Elemento SVG

Gráficos Escalables en Formato Vectorial (Scalable Vector Graphics) es un formato que usa texto para definir imágenes. Cada imagen SVG se define con código similar al de HTML. El código SVG se puede incluir directamente dentro de cualquier documento de

HTML. Todos los navegadores leen SVG, con excepción de Internet Explorer 8 y versiones anteriores. SVG está basado en XML, por consiguiente notará que los elementos que no tienen etiquetas de cierre - se auto-cierran. Por ejemplo:

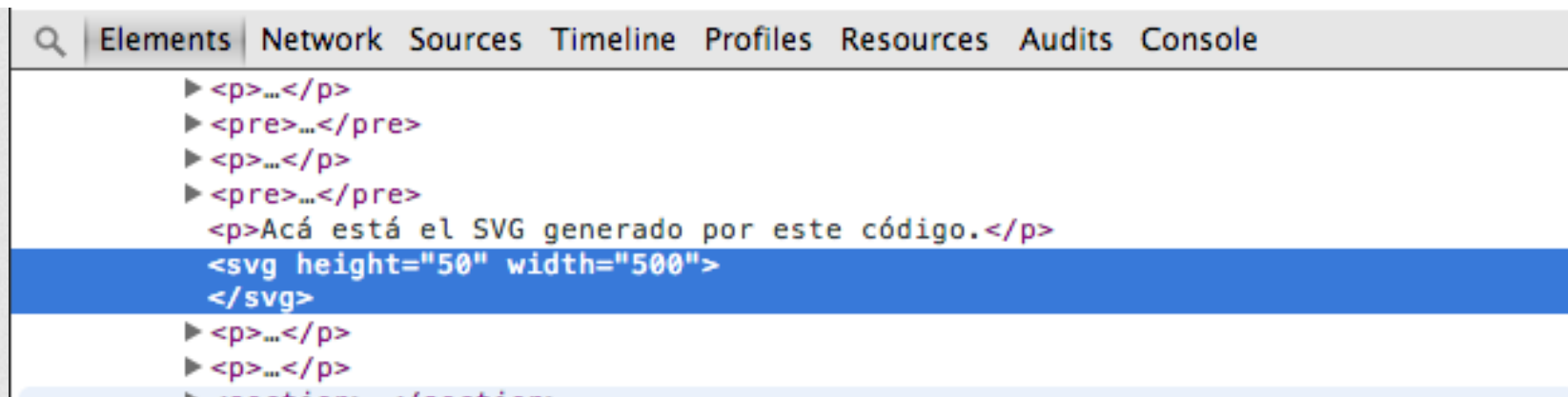
```
<element></element>  <!-- Usa una etiqueta de cierre -->
<element/>           <!-- Se auto-cierra -->
```

Antes de poder dibujar, se debe crear un elemento SVG. Piense en un elemento SVG como el lienzo donde se dibujan todos los gráficos. (En este contexto, SVG es conceptualmente similar elemento `canvas` de HTML). Por lo menos, se deben especificar los valores de ancho (`width`) y altura (`height`). Si éstos no se especifican, SVG utilizará todo el espacio que pueda dentro dentro del elemento que lo contiene.

```
<svg width="500" height="50">
</svg>
```

Acá está el SVG generado por este código.

Si no lo ve, puede usar el botón derecho y marcar en el espacio en blanco y luego seleccionar “Inspeccionar Elemento”. Su inspector de web mostrará algo similar a esto:



```
<p>...</p>
<pre>...</pre>
<p>...</p>
<pre>...</pre>
<p>Acá está el SVG generado por este código.</p>
<svg height="50" width="500">
</svg>
<p>...</p>
<p>...</p>
<section>...</section>
```

Tome nota de que ya existe un elemento `svg` (qué bien!) y que ocupa 500 píxeles horizontales y 50 píxeles verticales. No es que se muestre mucho por el momento (qué mal!).

También tome nota de que el navegador asumió que la medida por defecto es el *pixel*. Se especificó las dimensiones de 500 y 50 y no 500pxy 50px. Se habría podido especificar `px` explícitamente, o cualquier otro tipo de unidad de medida, incluyendo `em`, `pt`, `in`, `cm`, y `mm`.

Figuras Simples

Existen ciertos elementos visuales que se pueden incluir dentro de las etiquetas `svg`, incluyendo `rect`, `circle`, `ellipse`, `line`, `text`, y `path`.

Si usted tiene alguna familiaridad con la programación gráfica, reconocerá el sistema de coordenadas basado en píxeles, en el cual 0, 0 está ubicado en el la parte superior izquierda del espacio destinado a los dibujos. Al incrementar los valores `x`, estos se

mueven hacia la derecha, mientras que el incrementar los valores `y`, se mueven hacia abajo.

0,0 100,20 200,40

`rect` dibuja un rectángulo. Se usa `x` y `y` para especificar las coordenadas del costado superior izquierdo y `width` y `height` para especificar las dimensiones. Este rectángulo rellena todo el espacio de nuestro elemento SVG.

```
<rect x="0" y="0" width="500" height="50"/>
```

`circle` dibuja un círculo. Se usa `cx` y `cy` para especificar las coordenada del *centro* y `r` para especificar el radio. Este círculo queda centrado en la mitad del SVG de 500 pixeles de ancho, porque `cx` (“centro de x) tiene un valor de 250.

`ellipse` es similar, pero requiere dos radios, uno por cada eje. En vez de usar `r`, se usa `rx` y `ry`.

```
<ellipse cx="250" cy="25" rx="100" ry="25"/>
```

`line` dibuja una línea. Se usa `x1` y `y1` para especificar las coordenadas de extremo de la línea, y `x2` y `y2` para especificar las coordenadas del otro extremo. También es necesario especificar un color `stroke` para que se vea la línea.

```
<line x1="0" y1="0" x2="500" y2="50" stroke="black"/>
```

`text` despliega texto. Se usa `x` para especificar la posición del lado derecho y `y` para especificar la posición vertical de la línea de base del tipo.

Easy-peasy

`text` también hereda los estilos del fuente que se especifican en el CSS, específicamente los de los elementos del padre, si no se especifican explícitamente. (Próximamente se explicará en más detalle el uso de estilos para texto). Tome nota de cómo el estilo del texto de arriba tiene el mismo formato del texto de este párrafo. Es posible reemplazar ese formato de la siguiente manera:

```
<text x="250" y="25" font-family="sans-serif"
font-size="25" fill="gray">Easy-peasy</text>
```

Easy-peasy

También cabe anotar que cuando el elemento visual se corre sobre el costado del elemento SVG, éste lo corta. Se debe tener cuidado al usar `text` de tal manera que los descendientes no se corten (duele!). Puede ver lo que pasa cuando se define la línea base (`y`) en 50, lo mismo que la altura del SVG:

```
<text x="250" y="50" font-family="sans-serif"
font-size="25" fill="gray">Easy-peasy</text>
```

Easy-peasy

`path` se usa para dibujar formas más complejas que aquellas descritas anteriormente (como los límites de países en mapas) y será explicado en otra sección. Por el momento seguiremos trabajando con formas simples.

Estilos de los Elementos SVG.

El estilo por defecto de los SVGs es llenado en negro y sin línea. Si se necesita algo diferente, se tienen que aplicar estilos a los elementos. Algunas propiedades de SVG que se usan con frecuencia son:

- `fill` - Un valor para el color. Igual a que en CSS, los colores se pueden especificar así:
 - por nombre – orange
 - valor HEX – #3388aa o #38a
 - valor RGB – `rgb(10, 150, 20)`
 - RGB con transparencia alfa – `rgba(10, 150, 20, 0.5)`
- `stroke` – Un valor para el color.
- `stroke-width` – Una medida numérica (usualmente en pixeles).
- `opacity` – Un valor numérico entre 0.0 (completamente transparente) y 1.0 (completamente opaco).

Con `text`, se pueden usar estas propiedades, que funcionan de la misma forma que en CSS”

- font-family
- font-size

Haciendo otro paralelo con CSS, existen dos formas de aplicar estilos a un elemento SVG: directamente (en línea) como un atributo del elemento, o dentro de una regla de estilo de CSS.

Acá se muestran algunas propiedades de estilo aplicadas directamente como atributos a `circle`.

```
<circle cx="25" cy="25" r="22"  
fill="yellow" stroke="orange" stroke-width="5"/>
```

De otra manera, se pueden omitir los atributos de estilo y asignarle una clase a `circle` (como si fuera un elemento HTML)

```
<circle cx="25" cy="25" r="22" class="pumpkin"/>
```

y después incluir las reglas de `fill`, `stroke`, y `stroke-width` dentro de un estilo CSS asignado a esta nueva clase:

```
.pumpkin {  
  fill: yellow;  
  stroke: orange;  
  stroke-width: 5;  
}
```

El usar CSS tiene algunos beneficios obvios:

1. El estilo se especifica una vez y se puede aplicar a varios elementos.
2. El código CSS es usualmente más fácil de leer que el código en línea.
3. Por esas razones, tiende a ser más fácil mantener el código con el enfoque de CSS e igualmente es más fácil implementar cambios de diseño.

El usar CSS para aplicar estilos SVG, sin embargo, puede ser algo desconcertante para algunos. `fill`, `stroke`, y `stroke-width`, después de todo, no son propiedades de CSS. (Los equivalentes más cercanos en CSS son `background-color` y `border`.). Si en algo ayuda recordarse cuales reglas dentro de CSS son específicas a SVG, vale la pena incluir `svg` en los selectores:

```
svg .pumpkin {  
  /* ... */  
}
```


Capas y el Orden de Precedencia de los Dibujos

En SVG no existen las “capas” y no existe el concepto de profundidad. SVG no soporta la propiedad `z-index` de CSS, por eso las figuras solamente se pueden organizar en un plano x/y de dos dimensiones.

Sin embargo, si dibujamos varias figuras, se traslapan:

```
<rect x="0" y="0" width="30" height="30" fill="purple"/>
<rect x="20" y="5" width="30" height="30" fill="blue"/>
<rect x="40" y="10" width="30" height="30" fill="green"/>
<rect x="60" y="15" width="30" height="30" fill="yellow"/>
<rect x="80" y="20" width="30" height="30" fill="red"/>
```

El orden en que se incluyen los elementos en el código determina su orden (de profundidad). El cuadrado morado aparece de primero en el código y por eso se dibuja de primero. Luego, el cuadrado azul se dibuja “encima” del morado, luego el verde encima del azul, y así en adelante.

Toca pensar en las figuras de SVG como se dibujarían figuras en un lienzo. La pintura de pixeles que se aplica después oscurece lo que se haya dibujado antes e igualmente aparece como si estuviera “adelante”.

El tema de orden en el dibujo es importante cuando se tienen elementos visuales que nos deben ser bloqueados por otros. Por ejemplo, cuando se tienen etiquetas para los ejes o los valores en un diagrama de dispersión. Los ejes y las etiquetas se deben añadir al SVG de último, para que aparezcan encima de los demás elementos.

Transparencia

Las transparencia puede ser útil cuando existen elementos que se traslapan pero que deben ser visibles. También cuando se le debe quitar algo de importancia a algunos elementos y darle importancia a otros.

Existen dos formas de aplicar transparencia: usando un color RGB con alpha o asignado un valor de opaco.

Se puede usar `rgba()` en cualquier lugar donde se especifique un color, tal como `fill` o `stroke.rgba()` que esperan tres valores entre 0 y 255 para el rojo, verde y azul, más el valor de transparencia (alpha) que está entre 0.0 y 1.0.

```
<circle cx="25" cy="25" r="20" fill="rgba(128, 0, 128, 1.0)"/>
<circle cx="50" cy="25" r="20" fill="rgba(0, 0, 255, 0.75)"/>
<circle cx="75" cy="25" r="20" fill="rgba(0, 255, 0, 0.5)"/>
<circle cx="100" cy="25" r="20" fill="rgba(255, 255, 0, 0.25)"/>
<circle cx="125" cy="25" r="20" fill="rgba(255, 0, 0, 0.1)"/>
```

Cabe anotar que con `rgba()`, la transparencia se aplica independientemente a los colores de `fill` y `stroke`. Los círculos que se muestran a continuación tienen un `fill` de 75% opaco, y sus valores de `stroke` son de 25% opaco.

```
<circle cx="25" cy="25" r="20"  
  fill="rgba(128, 0, 128, 0.75)"  
  stroke="rgba(0, 255, 0, 0.25)" stroke-width="10"/>  
<circle cx="75" cy="25" r="20"  
  fill="rgba(0, 255, 0, 0.75)"  
  stroke="rgba(0, 0, 255, 0.25)" stroke-width="10"/>  
<circle cx="125" cy="25" r="20"  
  fill="rgba(255, 255, 0, 0.75)"  
  stroke="rgba(255, 0, 0, 0.25)" stroke-width="10"/>
```

Para aplicar transparencia a todo el elemento, se asigna el atributo `opacity`. Aquí están unos círculos completamente opacos

seguidos por los mismos círculos con valores de `opacity`.

Se puede aplicar `opacity` a un elemento que tenga los colores definidos con `rgba()`. Al hacer eso, las transparencias se multiplican. Los siguientes círculos usan los mismos valores RGBA para `fill` y `stroke`. El primer círculo debajo no tiene el elemento `opacity`, pero los otros dos sí:

```
<circle cx="25" cy="25" r="20"  
  fill="rgba(128, 0, 128, 0.75)"  
  stroke="rgba(0, 255, 0, 0.25)" stroke-width="10"/>  
<circle cx="65" cy="25" r="20"  
  fill="rgba(128, 0, 128, 0.75)"  
  stroke="rgba(0, 255, 0, 0.25)" stroke-width="10"  
  opacity="0.5"/>  
<circle cx="105" cy="25" r="20"  
  fill="rgba(128, 0, 128, 0.75)"  
  stroke="rgba(0, 255, 0, 0.25)" stroke-width="10"  
  opacity="0.2"/>
```

Mire como el tercer círculo tiene el valor `opacity` de 0.2 o 20%. Sin embargo, el relleno morado ya tiene un valor alpha de 0.75 o 75%. El área morada, entonces tiene una transparencia final de 0.2 por 0.75= 0.15 o 15%.

Siguiente : Despliegue de SVG →

Este tutorial fue traducido por Gabriel Coch

Todo el contenido fue desarrollado por y le pertenece a Scott Murray